Towards a Provenance-Aware Internet of Things (IoT) System

Ebelechukwu Nwafor [*], David Hill[*], Andre Campbell[*] and Gedare Bloom[*]
[*]Department of Electrical Engineering and Computer Science
Howard University, Washington, DC 20059
Email: ebelechukwu.nwafor@bison.howard.edu

Abstract—**The Internet of Things (IoT) offers immense benefits by enabling devices to leverage networked resources thereby making intelligent decisions. The numerous heterogeneous connected devices that exist throughout the IoT system creates new security and privacy concerns. Some of these concerns can be overcome through trust, transparency, and integrity, which can be achieved with data provenance. Data provenance, also known as data lineage, provides a history of transformations that occurs on a data object from the time it was created to its current state. Data provenance has been explored in the areas of scientific computing, business, forensic analysis, and intrusion detection. Data provenance can help in detecting and mitigating malicious cyber attacks. In this paper, we explore the integration of provenance within the IoT. We propose a provenance collection framework for IoT devices. We evaluate the effectiveness of our framework by looking at an application of provenance data by developing a prototype system for proof of concept.**

I. Introduction

The Internet of Things (IoT) is transforming commercial and industrial information technology all over the world. Heterogeneous devices are communicating with each other over a shared network (e.g internet). For example, you can automatically control the temperature of a house remotely though the use of a cell phone.

IoT offers immense benefits in the areas of home, industrial and vehicle automation by making operations which might require lots of interaction seamless and also preventing accidents which might be due to human error. However, with this unprecedented level of devices (things) communicating in the IoT, there has been an exponential increase in the number of devices connected to the internet. It is estimated that over 50 million devices will be connected to the internet by the year 2020. With the vast amounts of connected heterogeneous devices, security and privacy risks is increased. Rapid7 [1], an internet security and analytics organization, released a report highlighting vulnerabilities that exist on select IoT devices. In their report, they outline vulnerabilities in baby monitors which allowed intruders unauthorized access to devices whereby a malicious intruder can view live feeds from a remote location.

Most IoT devices are not initially developed with security in mind. This raises the complexity of including security after the device has been deployed. For instance some devices come with default passwords which might never be changed during its lifecycle. The amount of data generated from IoT devices requires stronger levels of trust which can be achieved through data provenance. Data provenance is a comprehensive history of activities that occurred on an entity from its origin to its current state. Provenance ensures integrity of data . Provenance has been applied in various domains such as scientific workflow for experiment reproducibility, information security as a form of access control and also for intrusion detection in mitigating malicious adversaries. Provenance ensure trust and integrity of data. For IoT devices (things) that produces lots of sensor-actuator data, a workflow representation of of how sensor data is generated can be used to depict dependency between sensor-actuator readings and devices/sensor information contained in the device. This information generated prove to be beneficial as a means for mitigating malicious intrusion or for scientific reproducability.

In this paper, we propose a provenance aware framework for IoT devices, in which provenance data is collected and modeled to represent dependencies between sensor-actuator readings and the various entities contained in the IoT architecture. Most of the interconnected heterogeneous devices (things) are embedded systems which require lightweight and efficient solutions as compared to general purpose systems. This requirement is attributed to the constrained memory and computing power of such devices.

The remaining portion of this paper is organized as follows: section 2 discusses background information on IoT definition, architecture, application domains and data provenance. Section 3 motivates the need for incorporating provenance to IoT with a use case scenario of an automated smart home. Section 3 talks about related work in provenance collection systems. Section 4 discusses implementation details for provenance collection framework. Section 5 talks about results and experiment analysis. Finally, in section 6 we conclude with future research directions.

II. Background

This section describes key concepts of data provenance, IoT characteristics, and provenance models. It also provides motivating example for the need for provenance collection via a use case.

A. Internet of Things

There is no standard definition of IoT. The notion of connected things is broad and encompasses several application domain making it hard to define however, researchers have tried to defined the IoT over the years.

In 1991, Mark Weiser [2], a pioneer of ubiquitous computing envisioned the notion of computing interleaved in our daily lives. Kevin Ashton [3], an early pioneer of IoT, in his presentation to Proctor & Gamble in on an idea of linking RFID in P&G's supply chain to the internet, defines the internet of things as devices in our everyday lives which can be identified connected to a network. These devices can learn from information gathered autonomously without human input which allows for improvements in waste reduction and overall standard of living. Buckley et al [4] defines the internet of things as a network of billions of machines communicating with each other. Gubbi et al [5] defines the IoT as an interconnection of sensing and actuating devices that allows data sharing across platforms through a centralized framework. We define (IoT) as follows:

The Internet of Things (IoT) is a network of heterogeneous devices with sensing and actuating capabilities communicating over the internet.

IoT has applications in home automation, smart health, automotive communication, machine to machine comminication, industrial automation. The notion of IoT has been attributed to smart devices. The interconnectivity between various heterogeneous devices allows for devices to share information in a unique manner. Analytics is a driving force for IoT. With analytics, devices can learn from user data to make smarter decisions. This notion of smart devices is seen in various commercial applications such as smartwatches, thermostats that automatically learns a user patterns. The ubiquitous nature of these devices make them ideal choices to be included in consumer products. IoT architecture represents a functional hierarchy of how information is disseminated across multiple hierarchies contained in an IoT framework; from devices which contain sensing and actuating capabilities to massive data centers (cloud storage). Knowing how information is transmitted across layers allows a better understanding on how to model the flow of information across actors contained in an IoT hierarchy. Figure 1 displays the IoT architecture and the interactions between the respective layers. IoT architecture consists of four distinct layers: The sensor and actuator layer, device layer, gateway layer and the cloud layer. The base of the architectural stack consist of sensors and actuators which gathers provenance information and interacts with the device layer. The device layer consists of devices (e.g mobile phones, laptops, smart devices) which are responsible for aggregating data collected from sensors and actuators. These devices in turn forwards the aggregated data to the gateway layer. The gateway layer routes and forwards data collected from the device later. It could also serve as a medium of temporary storage and data processing. The cloud layer is involved with the storage and processing of data collected from the gateway layer. Note that the resource constraints decreases up the architectural stack with the cloud layer having the most resources (memory, power computation) and the sensor-actuator layer having the least.

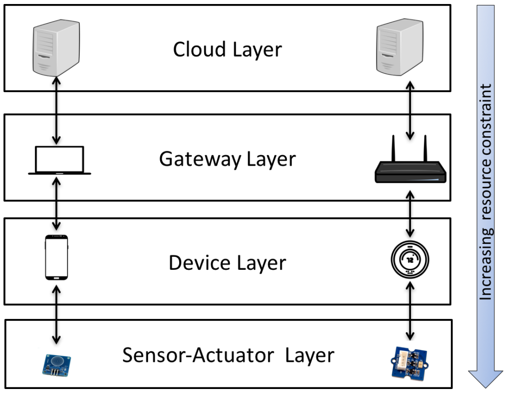

Fig. 1. IoT Architecture Diagram. The arrows illustrates the interaction between data at various layers on the architecture.

With the recent data explosion [22] due to the large influx in amounts of interconnected devices, information is disseminated at a fast rate and with this increase involves security and privacy concerns. Creating a provenance-aware system is beneficial to IoT because it ensures the trust and integrity of interconnected devices. Enabling provenance collection in IoT devices allows these devices to capture valuable information which enables backtracking in an event of a malicious attack.

B. Provenance-Aware IoT Device Use Case

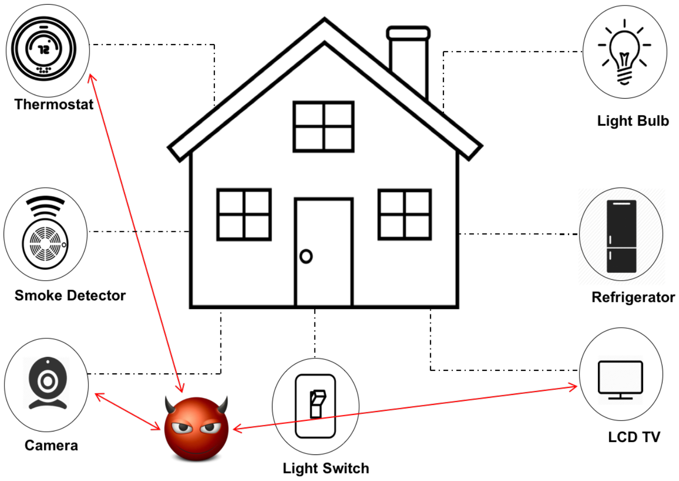

Fig. 2. Smart home use case Diagram

Consider a smart home as illustrated in Figure 2 that contains interconnected devices such as a thermostat which automatically detects and regulates the temperature of a room based on prior information of a user's temperature preferences, a smart lock system that can be controlled remotely and informs a user via short messaging when the door has been opened or is closed, a home security camera monitoring system, a smart fridge which sends a reminder when food products are low. In an event that a malicious intruder attempts to gain access to the smart lock system and security camera remotely, provenance information can be used to track the series of events to determine where and how a malicious attack originated. Provenance can also be used as a safeguard to alert of a possible remote or insider compromise thereby protecting against future or ongoing malicious attacks.

C. Data Provenance

The Oxford English dictionary [6] defines provenance as "the place of origin or earliest known history of something". An example of provenance can be seen with a college transcript. A transcript is the provenance of a college degree because it outlines all of the courses satisfied in order to attain the degree. In the field of computing, data provenance, also known as data lineage, can be defined as the history of all activities performed on entities from its creation to its current state. Cheney et al [7] describes provenance as the origin and history of data from its lifecycle. Buneman et al [8] describes provenance from a database perspective as the origin of data and the steps in which it is derived in the database system. We formally define provenance as follows: Data provenance of an entity is a comprehensive history of activities that occur on that entity from its creation to its present state.

Provenance data is represented as an acyclic graph which denotes causal relationship and dependencies between entities.

1) Provenance Characteristics:

- Who: Provides information on activities made to an entity. Knowing the "who" characteristic is essential because it maps the identity of modification to a particular data object. An example of "who" in an IoT use case is a sensor device identifier.
- Where: Location information at which data transformation was made. This provenance characteristic could be optional since not every data modification contains location details. An example is a GPS coordinate.
- When: The time at which data transformation occurred. This is an essential provenance characteristic. Being able to tell the time of a data transformation allows for tracing data irregularities. An example of this characteristic is a timestamp that denotes when sensor data was read.
- What: The transformation is applied on a data object. A use case for IoT can be seen in the various operations (create, read, update, and delete) that could be performed on a data object.

2) Differentiating Provenance, Log data, and Metadata : Provenance has often been seen used interchangably to define log data, and metadata. While some overlap exists between the three, some differences exists. Log data contains information about the activities of an operating system or processes. Log data can be used as provenance because It contains data trace specific to an application domain. Log files might contain unrelated information such as error messages, warnings which might not be considered as provenance data. Provenance allows for specified collection of information that relates to the change of the internal state of an object. In summary, log data could provide insight to what provenance data to collect. On the other hand, metadata contains descriptive information about data. Metadata can be considered as provenance when there exists a relationship between objects and they explain the transformation that occurs. In summary, metadata and provenance are not the same, however an overlap exists. Metadata contains valuable provenance information but not all metadata is provenance information.

D. Model for representing provenance for IoT

In order to represent provenance in an IoT architecture, we need to satisfy provenance characteristics (i.e who, where, when, and what of data transformations).

Provenance of sensor and actuator readings in an IoT device should outline the dependency relationship between all entities responsible for producing such readings . Provenance can be represented using a modeling language with ontological representations such as PROV-DM which is represented serialized in three format: XML, JSON and RDF. This model displays data dependency between agents and entities. We propose an alignment to the PROV-DM which contains information such as sensor readings, device name, and device information and give a detailed description of the alignment with use cases. Details on the provenance-sensor data model alignment is discussed in section IV.

1) Provenance Data Model (Prov-DM): PROV-DM [9] is a model that conforms to PROV Ontology (PROV-O), a W3C standard that is used to depict dependencies between entities, activities and agents (digital or physical). It creates a common model that allows for interchange of provenance information between heterogeneous devices. PROV-DM is a predecessor of the Open Provenance model (OPM), a result of a meeting at the International Provenance and Annotation Workshop (IPAW) 2006. OPM was created to address the need of allowing a centralized model for representing provenance data amongst various applications. PROV-DM contains two major components: types and relations.

Types:

- Entity: An entity is a physical or digital object. An example of an entity is a file system, a process, or an motor vehicle. An entity may be physical or abstract.
- Activity: An activity represents some form of action that occurs over a time frame. Actions are acted upon by an agent. An example of an activity is a process opening a file directory, Accessing a remote server.
- Agent: An agent is a thing that takes ownership of an entity, or performs an activity. An example of an agent is a person, a software product, or a process.

The figure below illustrates the various types contained in PROV-DM and their representation. Entities, activities and agents are represented by oval, rectangle and hexagonal shapes respectively.

Relations:

- wasGeneratedBy: Signifies the creation of an entity by an activity.
- used: Denotes that the functionality of an entity has been adopted by an activity.

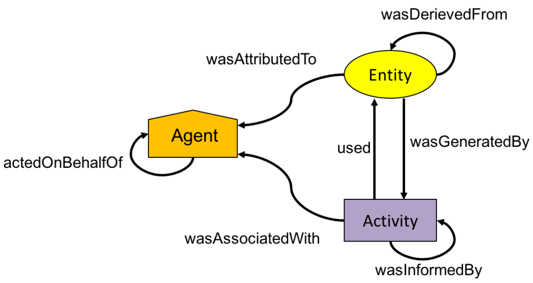

Fig. 3. Prov-DM respresentation showing various types contained in the model (Entity, Activity, and Agent)

- wasInformedBy: Denotes a causality that follows the exchange of two activities.
- wasDerivedFrom: Represents a copy of information from an entity.
- wasAttributedTo: Denotes relational dependency between an entity and an agent when the activity that created the agent is unknown.
- wasAssociatedWith: Denotes a direct association to an agent for an activity that occurs. This indicates that an agent plays a role in the creation or modification of the activity.
- actedOnBehalfOf: Denotes assigning authority to perform a particular responsibility to an agent.

III. Related Work

There has been a considerable amount of work done on data provenance collection [10]–[13]. Some of the work done has been focused on databases, sensor networks, scientific workflow systems and file system provenance but so far little attention has been given to provenance in IoT.

Muniswamy Reddy et al [14] developed a provenance collection system that tracks system-level provenance of the Linux file system. Provenance information is stored in the same location as the file system for easy accessibility, backup, restoration, and data management. Provenance information is collected and stored in the kernel space. PASS is composed of 3 major components: provenance collector, provenance storage,and provenance query. The collector keeps track of system level provenance. It intercepts system calls which are translated into provenance data and initially stored in memory as inode cache. Provenance data is then transferred to a file system in an kernel database, BerkleyDB. This database maps key value pairs for provenance data for fast index look up. Our approach looks at data provenance with data pruning as a key requirement since we are dealing with devices with limited memory and storage capabilities. We employ a policy based approach which allows provenance pruning by storing what provenance data is considered important to a specific organization. Also,unlike PASS, our system collects not just system level provenance but also application level provenance.

Bates et al. [15] developed system level provenance collection framework for the Linux kernel using Linux Provenance Modules(LPM), this framework tracks system level provenance such as interprocess communication, networking, and kernel activities. This is achieved by mediating access to kernel objects using Linux Security Model(LSM) which is a framework that was designed for providing custom access control into the Linux kernel. It consists of a set of hooks which executed before access decision is made. LSM was designed to avoid problem created by direct system call interception. The provenance information collected from the kernel space is securely transmitted to the provenance recorder in the user space. HiFi contains three components: provenance collector, provenance log and provenance handler. The collector and log are contained in the kernel space while the handler is contained in the user space. The log is a storage medium which transmits the provenance data to the user space. The collector uses LSM which resides in the kernel space. The collector records provenance data and writes it to the provenance log. The handler reads the provenance record from the log. This approach to collecting provenance data differs from our work since we focus on embedded systems and are concerned with input and output (I/O) data, which involves sensor and actuator readings. Additionally, HiFi deals with collecting system level events which might incur additional overhead when compared to collecting application level provenance. HiFi is engineered to work solely on the Linux operating system. Embedded systems that do not run on Linux OS will not be able to incorporate HiFi.

RecProv [16] is a provenance system which records user-level provenance, avoiding the overhead incurred by kernel level provenance recording. It does not require changes to the kernel like most provenance monitoring systems. It uses Mozilla rr to perform deterministic record and replay by monitoring system calls and non deterministic input. Mozilla rr is a debugging tool for Linux browser. It is developed for the deterministic recording and replaying of the Firefox browser in Linux. The provenance information generated is converted into PROV-JSON, and stored in Neo4j, a graph database for visualization and storage of provenance graphs.

Spillance et al [17] developed a user space provenance collection system, Storybook, which allows for the collection of provenance data from the user space thereby reducing performance overhead from kernel space provenance collection. This system takes a modular approach that allows the use of application specific extensions allowing additions such as database provenance, system provenance, and web and email servers. It achieves provenance capture intercepting system level events on FUSE, a file system and MySQL , a relational database. StoryBook allows developers to implement provenance inspectors these are custom provenance models which captures the provenance of specific applications which are often modified by different application(e.g web servers, databases). When an operation is performed on a data object, the appropriate provenance model is triggered and provenance data for that data object is captured. StoryBook stores provenance information such as open, close, read or write, application specific provenance, and causality relationship between entities contained in the provenance system. Provenance data is stored in key value pairs using Stasis and Berkely DB as the storage backend. It also allows the use of interoperable data specifications such as RDF to transfer data between various applications. Storybook allows for provenance query by looking up an inode in the ino hashtable. Collecting application level and kernel level events is similar to our approach of provenance collection, however, our approach integrates the use of a policy to eliminate noisy provenance data thereby allowing only relevant provenance to be stored.

Lim et al. [18] developed a model for calculating the trust of nodes contained in a sensor network by using data provenance and data similarity as deciding factors to calculate trust. The value of provenance signifies that the more similar a data value is, the higher the trust score. Also, the more the provenance of similar values differ, the higher their trust score. The trust score of a system is affected by the trust score of the sensor that forwards data to the system. Provenance is determined by the path in which data travels through the sensor network. This work differs from our approach since the authors focus on creating a trust score of nodes connected in a sensor network using data provenance and do not emphasize how the provenance data is collected. We are focused on creating a provenance aware system for sensor-actuator I/O operations which may be used to ensure trust of connected devices.

IV. Prov-Sensor Model Alignment

In this section, we describe the sensor-provenance model alignment, we discuss how sensor and actuator readings are converted to provenance using provenance ontological constructs.

Sensor data contains observation information such as temperature, location details which can be represented in various standardized data interchange formats (RDF, XML, JSON e.t.c). For our implementation, sensor data is generated in CTF format. Further information on CTF is described in section V. Trace data containing sensor-actuator readings are important components but do not depict dependency relationship when used alone. Trace data is transformed to provenance to depict relationship between entities contained in an IoT system. Provenance Ontology and sensor data can be further aligned for better representation of dependency relationships between sensor data.

A single sensor has the capability of collecting multiple trace data, t_d. A combination of trace data is regraded as an event. An example of events are temperature, location, humidity information e.t.c. We define an event for sensor s_1 as $e = \{td_1, ...td_n\}$ where td_1 is the first trace data collected by s_1 and td_n is the last trace data contained collected by the sensor.

$[sensor, actuator, device] \in \{prov : agent\}$
$[datapoint] \in \{prov : entity\}$

Using the PROV-O, in an IoT system containing multiple sensors, device and sensors information are represented as agents (prov:agents), the operation performed on sensor-actuator data (read, create, update) represents provenance activity (prov:activity). Events are represented as an entities (prov:entity). Each sensor-actuator data generated is represented as a tuple $< t, e, s_1, r_1 >$ where t represents a timestamp, e represents an event in which trace data is collected, s_1 represents sensor information and r_1 represents device information.

1) Example: Consider a humidity and temperature sensor s_1, connected to device r_1, a raspberry pi. Event, e, can be defined as $e = \{temperature, humidity\}$ therefore the tuple representation of trace data for raspberry pi r_1 with sensor s_1 is represented as $< t, \{temperature, humidity\}, s_1, r_1 >$. Each trace data generated is mapped to the PROV-O representation using the defined constructs specified above. Figure 4 further illustrates the concept of Prov-Sensor alignment using a graphical representation which contains a device connected to three sensors $s_1, s_2, and s_3$ with events $[e_1, e_2, ..., e_n]$.

Data is generated by three identical temperature sensors, s_1, s_2 and s_3. All sensors generate data in real-time. Trace data generated from devices are aligned with PROV-O. The graph depicts data dependency between the raspberry pi r_1, the three sensors, the activity performed by the sensors (the sensors create data in this case) and the events. Each event makes an edge with the preceding event. Edges between events are denoted with a dotted arrow . This represents time dependency between events. We define the graph of all of the events as follows:

For graph G= (V, E):
$V = \{e_1, ...e_n\}$
$E = \{(e_i, e_{i+1}) : e_i, e_{i+1} \in V, e_1 \neq e_{i+1}\}$

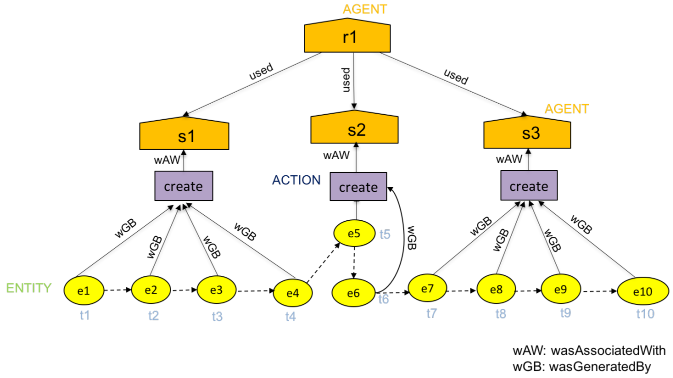

Fig. 4. Prov-Sensor Model Alignment

V. Provenance-Aware System Implementation

In this section, we outline components the provenance collection system and describe how provenance trace is collected across the IoT system. Figure 5 displays the system architecture of our approach. Sensor and actuator readings in the form of input and output (I/O) events are recorded by the tracer component. This component intercepts system level I/O

Algorithm 1: Prov-Sensor Alignment

```
Function trace2Prov (F)
    p ← createProvDocument()
    for $x \in F$ do
        if $s \notin p$ then
            s ← createAgent()
        if $d \notin p$ then
            d ← createAgent()
        e ← createEntity()
        c ← createActivity()
        if $x \notin p$ then
            x ← relateSensorToDevice()
        if $y \notin p$ then
            y ← relateActivityToSensor()
        z ← relateEntityToActivity()
```

events and produces trace information represented in Common Trace Format (CTF).

CTF Trace information is converted to provenance data in the PROV-DM IoT model and serialized to PROV-JSON. CTF conversion to PROV-DM is achieved using babeltrace. This conversion can happen at any layer of the IoT stack. Babeltrace is a plugin framework which allows the conversion of CTF traces into other formats. Trace or provenance data is securely transmitted to a gateway and later transmitted and stored in a cloud backend. Our backend of choice is Neo4j, a graph database for efficient storage, query and visualization of provenance data. CTF contains a mandatory stream known as metadata. Metadata contains information of other streams. It allows to parse a stream without specifying a layout.

1) Common Trace Format (CTF) : CTF allows traces to be generated by C/C++ applications. It encodes binary trace output information containing multiple streams of binary events such as I/O activity. CTF is composed of multiple streams of events. Each event can be broken into as multiple streams. Streams allows for fast processing since they do not have to be stored in disk before being sent over a network or processed in memory.

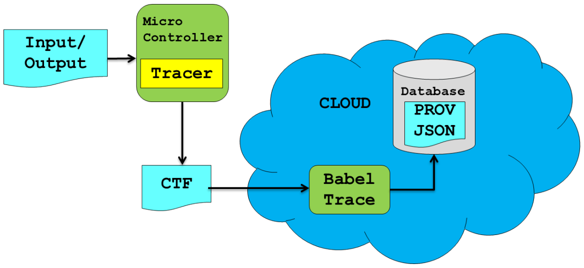

Fig. 5. System Architecture for Provenance Collection.

Our goal is to create a provenance-aware system which records I/O operations on data for devices connected in an IoT system. For our implementation, several tools and hardware components are utilized in the development of our prototype, outlined below:

- Raspberry Pi the microcontroller used to evaluate our approach. We choose Raspberry Pi because it is a representation of what can be found on an IoT gateway device and it has the capability to include custom hardware in programmable logic. Also, Raspberry Pi is a low cost, simple IoT demonstrator that was chosen for its high performance, onboard emulation, and IoT gateway projects can be programmed without additional need for hardware tools.
- Neo4j [1], a graph database which allows optimized querying of graph data. Since provenance represents causal dependencies, it is ideal to use a graph database to store the relationships between objects
- Babeltrace[2]: This is a trace converter tool for barectf. It contains plugins used to convert traces from one format into another (e.g CTF to other formats).
- barectf[3]: This is a light weight command line generator of C tracers used to generate bare metal application trace. Trace data collected is generated in CTF.
- yaml generator: A yaml generator is contained in a barectf. It creates yaml files. It contains configuration file contains information on what barectf application needs in order to generate CTF trace output. This consist of configuration settings such as an application trace stream, packet type, payload type and size.

VI. Conclusion

In this paper, we motivate the need for integrating provenance into the IoT system. We propose a provenance collection framework that provides provenance collection capabilities for devices in an IoT system.

VII. Future Work

Some of the proposed future work are outlined below:

- Provenance collection raises privacy issues. How do we ensure that the vast amount of data collected is not invasive to privacy.
- Securing Provenance: Proper encryption and authentication techniques [19] are needed to ensure the confidentiality, and integrity of provenance data.

Acknowledgment

This research has been supported in part by US National Science Foundation (CNS grant No:1646317), and by a generous gift from Leidos. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of NSF or Leidos.

[1] https://neo4j.com/product/

[2] http://diamon.org/babeltrace/

[3] https://github.com/efficios/barectf

References

[1] M. Stanislav, "Hacking iot: A case study on baby monitor exposures and vulnerabilities," https://www.rapid7.com/docs/Hacking-IoT-A-Case-Study-on-Baby-Monitor-Exposures-and-Vulnerabilities.pdf, Sep 2015, accessed: 2016-10-01.

[2] M. Weiser, "The computer for the 21st century," *Scientific american*, vol. 265, no. 3, pp. 94–104, 1991.

[3] K. Ashton, "That internet of things thing," *RFiD Journal*, vol. 22, no. 7, pp. 97–114, 2009.

[4] L. Yan, Y. Zhang, L. T. Yang, and H. Ning, *The Internet of things: from RFID to the next-generation pervasive networked systems*. CRC Press, 2008.

[5] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems*, vol. 29, no. 7, pp. 1645–1660, Sep. 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167739X13000241

[6] Rationality., *The Cambridge Dictionary of Philosophy*. Cambridge University Press, 1999.

[7] J. Cheney, L. Chiticariu, and W.-C. Tan, "Provenance in Databases: Why, How, and Where," *Found. Trends databases*, vol. 1, no. 4, pp. 379–474, Apr. 2009. [Online]. Available: http://dx.doi.org/10.1561/1900000006

[8] P. Buneman, S. Khanna, and T. Wang-Chiew, "Why and Where: A Characterization of Data Provenance," in *Database Theory ICDT 2001*, ser. Lecture Notes in Computer Science, J. V. d. Bussche and V. Vianu, Eds. Springer Berlin Heidelberg, Jan. 2001, no. 1973, pp. 316–330, dOI: 10.1007/3-540-44503-X_20. [Online]. Available: http://link.springer.com/chapter/10.1007/3-540-44503-X_20

[9] "PROV-DM: The PROV Data Model," https://www.w3.org/TR/prov-dm/, accessed: 2016-10-01. [Online]. Available: https://www.w3.org/TR/prov-dm/

[10] P. Macko and M. Seltzer, "A general-purpose provenance library," in *Proceedings of the 4th USENIX Conference on Theory and Practice of Provenance*, ser. TaPP'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 6–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=2342875.2342881

[11] A. Bates, D. J. Tian, K. R. Butler, and T. Moyer, "Trustworthy whole-system provenance for the linux kernel," in *24th USENIX Security Symposium (USENIX Security 15)*, 2015, pp. 319–334.

[12] E. Gessiou, V. Pappas, E. Athanasopoulos, A. D. Keromytis, and S. Ioannidis, "Towards a Universal Data Provenance Framework Using Dynamic Instrumentation," in *Information Security and Privacy Research*, ser. IFIP Advances in Information and Communication Technology, D. Gritzalis, S. Furnell, and M. Theoharidou, Eds. Springer Berlin Heidelberg, Jun. 2012, no. 376, pp. 103–114, dOI: 10.1007/978-3-642-30436-1_9. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-642-30436-1_9

[13] K.-K. Muniswamy-Reddy, P. Macko, and M. Seltzer, "Provenance for the Cloud," in *Proceedings of the 8th USENIX Conference on File and Storage Technologies*, ser. FAST'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 15–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1855511.1855526

[14] K.-K. Muniswamy-Reddy, D. A. Holland, U. Braun, and M. Seltzer, "Provenance-aware Storage Systems," in *Proceedings of the Annual Conference on USENIX '06 Annual Technical Conference*, ser. ATEC '06. Berkeley, CA, USA: USENIX Association, 2006, pp. 4–4. [Online]. Available: http://dl.acm.org/citation.cfm?id=1267359.1267363

[15] D. J. Pohly, S. McLaughlin, P. McDaniel, and K. Butler, "Hi-Fi: Collecting High-fidelity Whole-system Provenance," in *Proceedings of the 28th Annual Computer Security Applications Conference*, ser. ACSAC '12. New York, NY, USA: ACM, 2012, pp. 259–268. [Online]. Available: http://doi.acm.org/10.1145/2420950.2420989

[16] Y. Ji, S. Lee, and W. Lee, "RecProv: Towards Provenance-Aware User Space Record and Replay," in *Provenance and Annotation of Data and Processes*, ser. Lecture Notes in Computer Science, M. Mattoso and B. Glavic, Eds. Springer International Publishing, Jun. 2016, no. 9672, pp. 3–15, dOI: 10.1007/978-3-319-40593-3_1. [Online]. Available: http://link.springer.com/chapter/10.1007/978-3-319-40593-3_1

[17] R. Spillane, R. Sears, C. Yalamanchili, S. Gaikwad, M. Chinni, and E. Zadok, "Story Book: An Efficient Extensible Provenance Framework," in *First Workshop on the Theory and Practice of Provenance*, ser. TAPP'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 11:1–11:10. [Online]. Available: http://dl.acm.org/citation.cfm?id=1525932.1525943

[18] H.-S. Lim, Y.-S. Moon, and E. Bertino, "Provenance-based Trustworthiness Assessment in Sensor Networks," in *Proceedings of the Seventh International Workshop on Data Management for Sensor Networks*, ser. DMSN '10. New York, NY, USA: ACM, 2010, pp. 2–7. [Online]. Available: http://doi.acm.org/10.1145/1858158.1858162

[19] R. Hasan, R. Sion, and M. Winslett, "The case of the fake picasso: Preventing history forgery with secure provenance," in *Proccedings of the 7th Conference on File and Storage Technologies*, ser. FAST '09. Berkeley, CA, USA: USENIX Association, 2009, pp. 1–14. [Online]. Available: http://dl.acm.org/citation.cfm?id=1525908.1525909